笔记：

docker

docker的运作方式

docker是基于go语言开发的云开源项目。诞生 于DotCloud 2013年。Docker的主要目标是：Build，Ship and Run Any App，Anywhere 等生命周期的管理，达到应用组件级别的一次封装，到处运行。

Docker 是利用一个更好层次的控制工具，对进程进行封装隔离，是属于操作系统层面的虚拟化技术。隔离的进程独立于其他的进程。

Linux Container = Namespace + Cgroup

1. namespace 命名空间，主要做访问隔离
2. Cgroup 是control group的简称，又称为控制组，做资源控制。

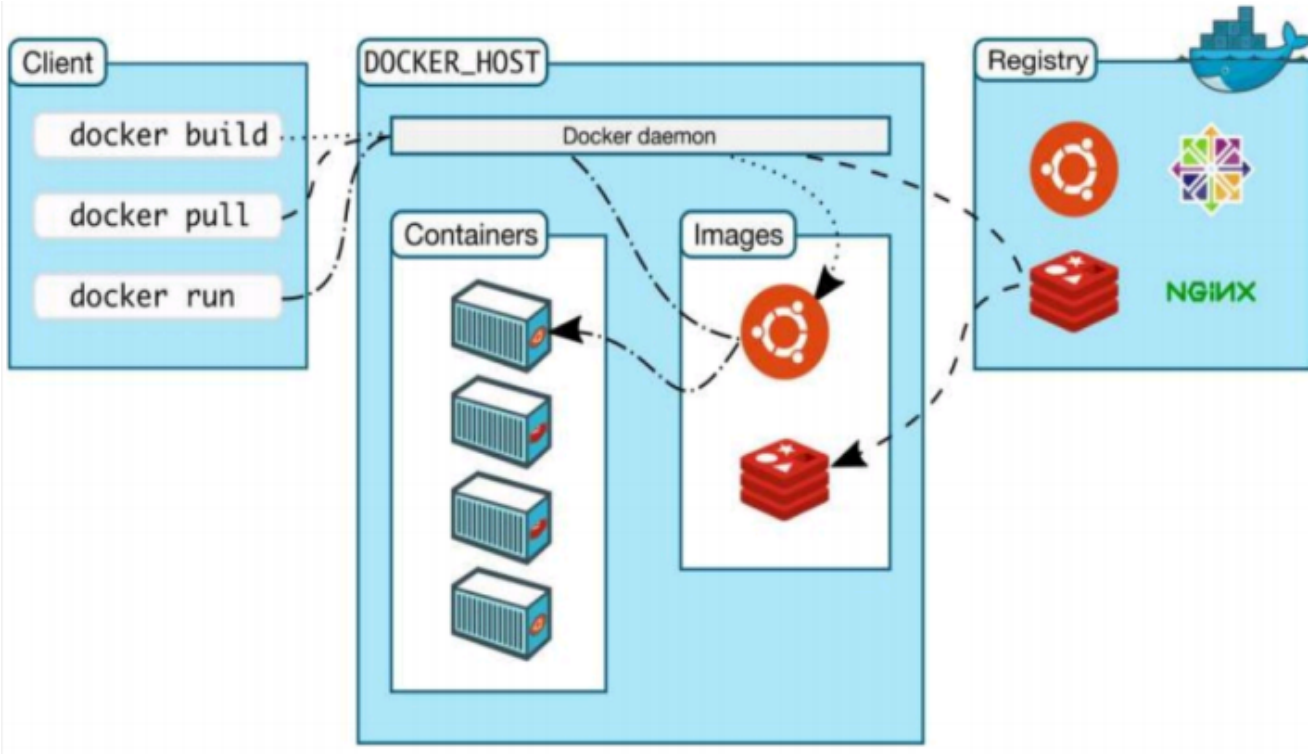
hypervisor层，容器相比虚拟机，省去了hypervisor层的开销。

hypervisor层：虚拟机监视器，用来建立与执行虚拟机的软件、固件或硬件。一种运行在基础物理服务器和操作系统之间的中间软件，可以运行多个操作系统和应用共享硬件。

容器和虚拟机的比较

特性	容器	虚拟机
启动速度	秒级	分钟级
硬盘使用	一般MB	一般 GB
性能	接近原生	弱于
系统支持量	单机支持上百容器	一般几十个
隔离性	安全隔离	完全隔离

docker 的运行



docker镜像命令

1. docker images 列出所有镜像
2. docker search TERM 搜索容器镜像
3. docker pull/push IMAGE 下载上传镜像
4. docker rmi IMAGE 删除镜像
5. docker commit CONTAINER IMAGE 建立容器镜像
6. docker build -t REPO:TAG PATH 建立容器镜像
7. docker tag IMAGE:TAG NEWIMAGE 给镜像打标签

Dockerfile 语法

- FROM 指定基础镜像
- MAINTAINER 指明镜像的作者和邮箱
- RUN 在新镜像内执行的命令
- COPY 将主机的文件复制到镜像内，不做解压
- ADD 类似 COPY，但是做解压文件的操作
- EXPOSE 暴露镜像的端口
- WORKDIR 构建镜像时，指定镜像工作目录
- USER 指定镜像以什么用户去执行
- VOLUME # 不建议使用，请忽略
- CMD 容器启动时需要执行的命令
- ENTRYPOINT 作用和 CMD 一样

CMD 和 ENTRYPOINT 的区别：

CMD 的命令会被 docker run 的命令覆盖但是 ENTRYPOINT 不会

Dockerfile 示例

```
FROM ubuntu
MAINTAINER sunhao sunhaoc@si-tech.com.cn
WORKDIR /usr/local/docker
ADD temp.zip ./add/
COPY temp.zip ./copy/
EXPOSE 22
RUN groupadd -r sunhaoc && useradd -r -g
sunhaoc  sunhaoc
USER vector4wang
ENTRYPOINT ["/bin/bash"]
```

k8s

kubernetes 是google开源的容器集群管理系统，其提供应用部署、维护，扩展机制等功能。利用kerbernets能够方便管理跨及其运行容器应用。k8s支持在平台上运行，也支持在物理主机上运行。

使用 Docker 对应用程序包装 (package) 、实例化 (instantiate) 、运行 (run)

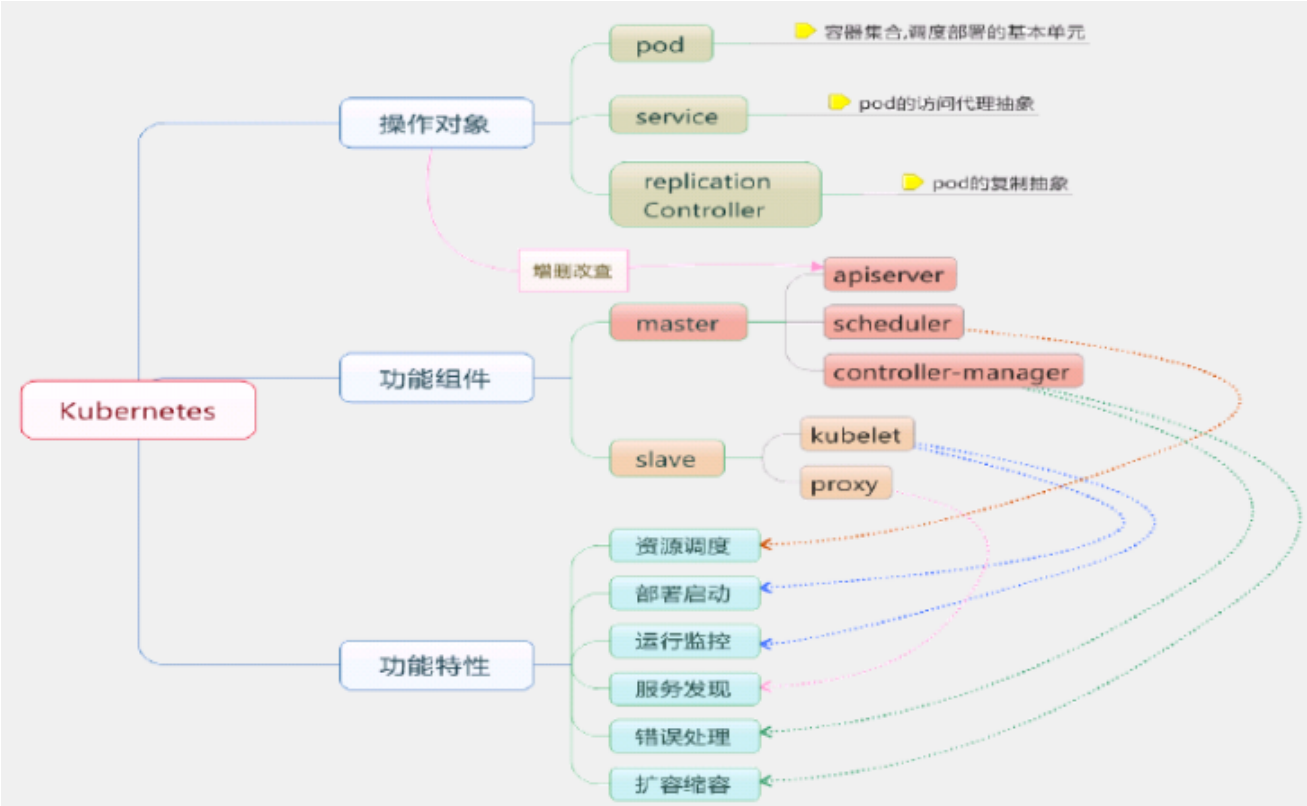
基于容器的应用部署、维护和滚动升级、以及自动伸缩

以集群的方式运行、管理跨机器的容器

解决 Docker 跨机器容器之间的通讯问题

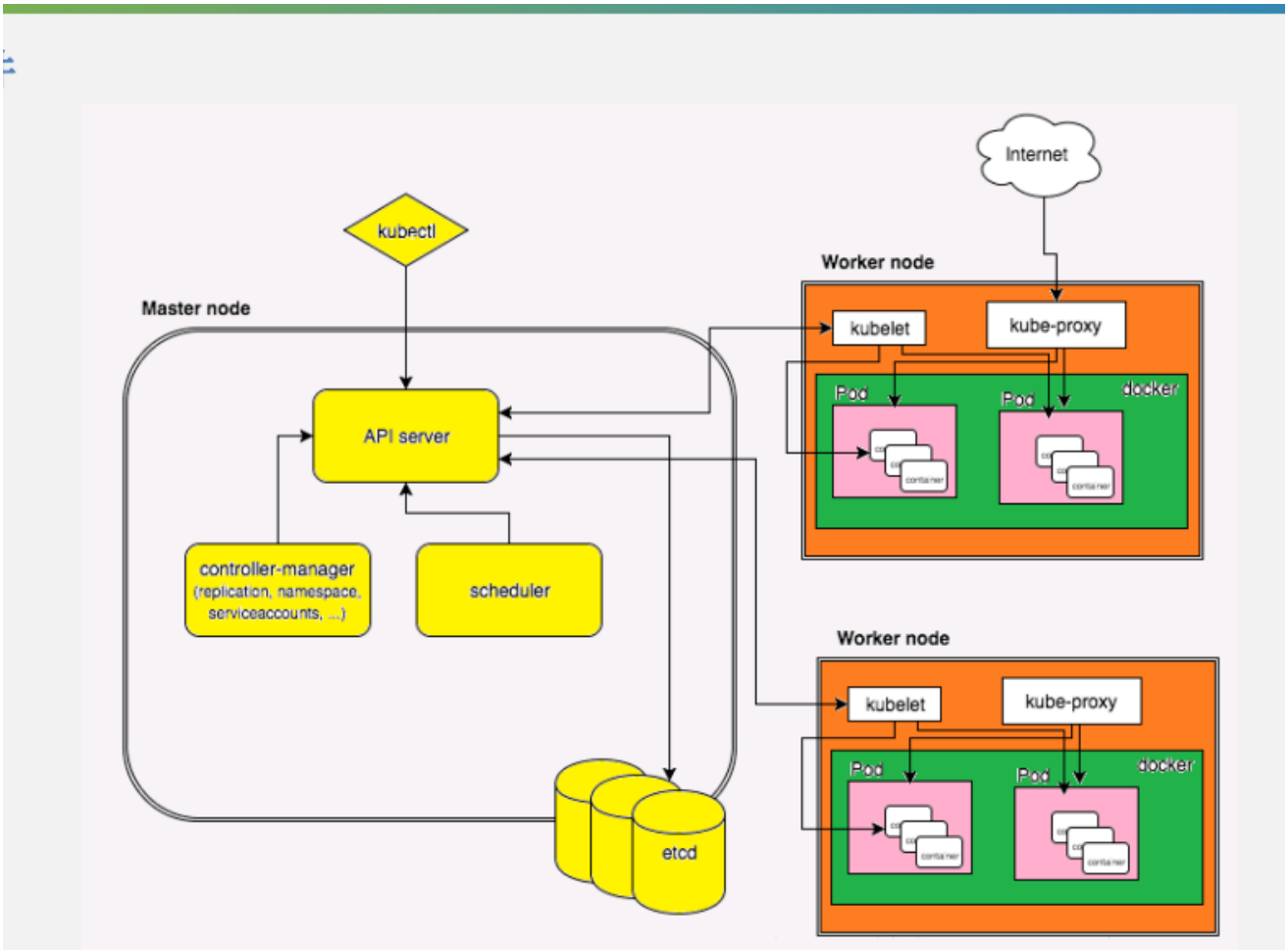
Kubernetes 的自我修复机制使得容器集群总是运行在用户期望的状态

k8s架构



是容器管理系统，本质是基于容器技术的mini-Paas平台

k8s组件



1. APISERVER 提供rest擦欧总接口
2. controller-manager执行大部分集群层次功能，执行生命周期功能（命名空间创建，声明周期 、事件垃圾手机等）。执行api业务逻辑。控制管理提供自愈、扩容、应用声明周期绑定和提供
3. schedduler组件为容器自动选择运行的主机。
4. kube-Proxy:每一个 woker 节点都需要运行一个 kube-proxy 守护进程，它能够按需为 Service 资源对象生成 IPTABLES 或 IPVS 的规则，从而捕获访问当前 Service 的 ClusterIP 的流量，并将其转发至正确的后端 Pod 对象。
5. Container Runtime Container-runtime 负责下载镜像和运行容器。 K8s 本身并不提供容器运行时环境，但提供了接口，可以插入所选择的

   容器运行时的环境。目前， Kubernetes 支持的容器运行时环境至少包括 Docker 、 RKT 、 CRI-O 和 Fraki 等。 Kubelet 作为客户端与 container-runtime 进行通信。

命令行方式部署

kubectl create -f file.yaml 或者 kubectl apply -f file.yaml

运维问题

k8s service创建之后，无法访服务?

- 线查看相关endpoint是否也被创建出来
- 确认endpoint后，在确认网络是否可达

- 同时查看网络策略是否做了配置

访问某个pod时，网络不可达?

- 测试该宿主主机的其他pod是否可达
- 如果不可达，确认该宿主主机的网络插件是否正常
- 如果可达，确认相关网络策略

访问nodeport的service时，有的主机能通，有的不能?

- 先确认不通的主机网络插件是否有异常
- 查看防火墙是否为deny

Pod 里运行的程序，无法写文件?

- 指定文件属主： pod.spec.securityContext. fsGroup

Pod 运行成功后，用户却都是 root ?

- 可以通过在写 Dockerfile 时，指定 USER
- pod.spec.securityContext. runAsUser 也可以指定

Pod 创建成功了，但是在 k8s 集群却没有发现新建 POD ?

一般这种问题，可能有多种情况导致的:

- 先确认 k8s 集群是否是正常的
- 再查看 deployment 或者 statefulset 的信息，通
- kubectl describe 的命令 , 可以获取信息提示
- 检查该 namespace 的相关资源配额，是否已经达到最

大值